Appl. No. 09/863,486
Amdt. dated April 18, 2005
Reply to Office action of January 18, 2005

Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A system ~~for processing multi-agent cooperative transactions~~, comprising:

a plurality of enterprises having agents that process multi-agent cooperative business transactions, wherein each agent is configured to autonomously determine whether to complete a transaction;

~~a)~~ a failure detector for detecting whether a failure is an inter-enterprise failure or an intra-enterprise failure;

~~b)~~ an intra-enterprise failure handler coupled to the failure detector for performing failure recovery for intra-enterprise failures; and

~~c)~~ an inter-enterprise failure handler coupled to the failure detector for performing failure recovery for inter-enterprise failures.

2. (Currently amended) The system of claim 1 wherein the intra-enterprise failure handler includes

a scope determination module for identifying ~~the~~a failure recovery scope; and

a top-down logical undo module coupled to the scope determination module for undoing sub-transactions in ~~the~~an identified scope in a top-down manner.

3. (Original) The system of claim 2 wherein the scope determination module terminates at a closest ancestor of the failed transaction that is one of non-vital and associated with a contingency transaction.

4. (Currently amended) The system of claim 2~~1~~ wherein in the scope determination module,

~~when~~if the~~a~~ failed transaction is non-vital to its parent, continue~~s~~ing with the parent transaction;

~~when~~if a failed sub-transaction is determined to be a result of agent malfunction, the parent transaction delegate~~s~~~~ing~~ the sub-transaction to another agent; and

~~when~~if the~~a~~ failed transaction is associated with a contingency transaction, re-trie~~s~~~~ying~~ the contingency transaction.

5. (Currently amended) The system of claim 1 wherein the inter-enterprise failure handler includes

a scope determination module for identifying ~~the~~a failure recovery scope; and

a top-down logical undo module coupled to the scope determination module for undoing sub-transactions in ~~the~~an identified scope in a top-down manner.

6. (Currently amended) The system of claim 5 wherein in the top-down logical undo module,

~~when~~if the~~a~~ transferred-~~in~~ transaction has not started, ~~then~~ does not perform~~ing~~ failure handling on the transferred-~~in~~ transaction;

~~when~~if the~~a~~ transferred-~~in~~ transaction is in progress, ~~then~~ determine~~s~~~~ing~~ the~~a~~ rollback root of the transferred-~~in~~ transaction; and

~~when~~if the~~a~~ transferred-~~in~~ transaction has completed, ~~then~~ compensate~~s~~~~ing~~ for the transferred-~~in~~ transaction.

7. (Currently amended) The system of claim 6~~1~~ wherein in the top-down logical undo module,

~~when~~if the rollback root of the transferred-~~in~~ transaction is in progress, ~~then~~ utilize~~s~~~~ing~~ the rollback root as the root of recovery;

whenif the rollback root of the transferred-in transaction has committed to one of a parent and an ancestor, the parent is in-progress, and the rollback root of the transferred-in transaction is one of non-vital and associated with a contingency transaction, then-utilizesing thea parent of the rollback root as the root of recovery; and

whenif the rollback root of the transferred-in transaction and thea parent of the rollback root have committed, then-determinesing thea highest committed ancestor of the rollback root of the transferred-in transaction and determinesing whether the rollback root of the highest committed ancestor is in progress;

whenif the highest committed ancestor is in progress, the highest committed ancestor of the rollback root of the transferred-in transaction is utilized as an extended rollback root;

whenif the highest committed ancestor is not in progress, the parent of the highest committed ancestor of the rollback root of the transferred-in transaction is utilized as an extended rollback root.

8. (Withdrawn) A method of processing multi-agent cooperative transactions comprising the steps of:

a) a first transaction waiting for a commit notification; the first transaction including a dependency on a second transaction;

b) the second transaction sending a commit notification; and

c) the first transaction receiving the commit notification and committing only after receipt of the commit notification.

9. (Withdrawn) The method of claim 8 wherein the dependency is a start dependency; and wherein the first transaction does not initiate processing until the second transaction is initiated.

10. (Withdrawn) The method of claim 8 wherein the dependency is a failure dependency; wherein the first transaction aborts when the second transaction aborts.

11. (Withdrawn) The method of claim 8 wherein the dependency between the first transaction and a second transaction is a mutual settle dependency; and wherein the first transaction and the second transaction can commit only when the first transaction and the second transaction have reached agreement on a contract.

12. (Withdrawn) The method of claim 8 wherein the dependency between the first transaction and a second transaction is a mutual abort-compensate dependency; and wherein when a first transaction fails, then the second transaction is either aborted or compensated for.

13. (Withdrawn) The method of claim 8 wherein sending a commit notification employs a point-to-point communication.

14. (Withdrawn) The method of claim 8 wherein sending a commit notification employs a multicast communication.

15. (Withdrawn) The method of claim 8 wherein the step of the first transaction receiving the commit notification and committing only after receipt of the commit notification includes

committing to a first database associated with the first transaction.

16. (Withdrawn) The method of claim 8 wherein a centralized coordinator for controlling the cooperation between transaction is not needed.

17. (Currently amended) A method ~~for processing failures in multi-agent cooperative transactions~~, comprising:

starting a multi-agent cooperative business transaction between software agents of at least two enterprises;

a) ——detecting whether a failure is an inter-enterprise failure or an intra-enterprise failure;

b) ——when the failure is an intra-enterprise failure, performing failure recovery for the intra-enterprise failure; and

c) ——when the failure is an inter-enterprise failure, performing failure recovery for the inter-enterprise failure.

18. (Original) The method of claim 17 wherein the step of when the failure is an intra-enterprise failure, performing failure recovery for the intra-enterprise failure includes

identifying the failure recovery scope; and

undoing sub-transactions in the identified scope in a top-down manner.

19. (Original) The method of claim 17 wherein the step of when the failure is an inter-enterprise failure, performing failure recovery for the inter-enterprise failure includes

identifying the failure recovery scope; and

undoing sub-transactions in the identified scope in a top-down manner.

20. (Original) The method of claim 19 wherein identifying the failure recovery scope includes the step of terminating at a closest ancestor of the failed transaction that is one of non-vital and associated with a contingency transaction.

21. (New) A system, comprising:

a plurality of software agents that process multi-agent cooperative business transactions, wherein each agent is configured to autonomously make decisions that affect completion of a business transaction; and

a failure detector that detects if a business transaction fails; and

a failure handler coupled to the failure detector for performing failure recovery.

22. (New) The system of claim 21 wherein the plurality of software agents communicate based on a peer-to-peer mechanism that enables an agent to delegate a task to another agent and to copy information that is to be delegated to said other agent.

23. (New) The system of claim 21 wherein the plurality of software agents communicate based on a peer-to-peer mechanism that enables each agent to log hierarchies of related transactions and to recover a logged transaction if a failure occur.

24. (New) The system of claim 21 wherein the plurality of software agents communicate based on a peer-to-peer mechanism that includes a preliminary commit stage and a final commit stage.

25. (New) The system of claim 21 wherein the failure handler aborts a transaction if the transaction is determined to be non-vital to an associated parent transaction.

26. (New) The system of claim 21 wherein, if a failure of a child transaction is determined to result from a malfunction of a first agent, the failure handler delegates a transaction from the first agent to a second agent.

27. (New) The system of claim 21 wherein the failure handler replaces a failed transaction with a contingency transaction.

28. (New) The system of claim 21 further comprising,

a first database that stores transaction data related to a first enterprise; and

a second database that stores transaction data related to a second enterprise;

wherein at least one agent is committed to the first database and at least one agent is committed to the second database.